M11-L1 Problem 3

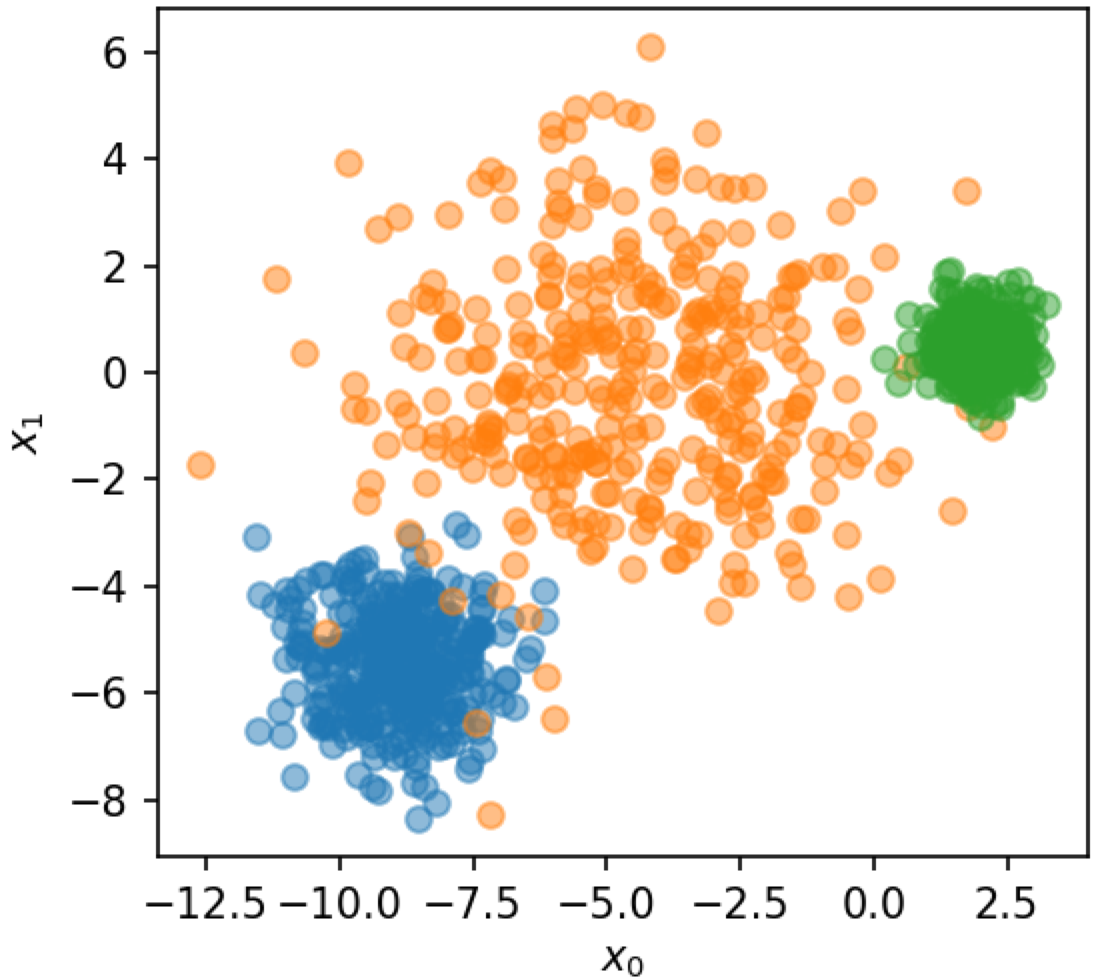
In this problem you will use the `sklearn` implementation of hierarchical clustering with three different linkage criteria `('single', 'complete', 'average')` to clusters two datasets: a "blob" shaped dataset with three classes, and a concentric circle dataset with two classes.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt

from sklearn.datasets import make_blobs, make_circles
from sklearn.cluster import AgglomerativeClustering

## DO NOT MODIFY
def plotter(x, labels = None, ax = None, title = None):
    if ax is None:
        _, ax = plt.subplots(dpi = 150, figsize = (4,4))
        flag = True
    else:
        flag = False
    for i in range(len(np.unique(labels))):
        ax.scatter(x[labels == i, 0], x[labels == i, 1], alpha = 0.5)
    ax.set_xlabel('$x_0$')
    ax.set_ylabel('$x_1$')
    ax.set_aspect('equal')
    if title is not None:
        ax.set_title(title)
    if flag:
        plt.show()
    else:
        return ax
```

First we will consider the "blob" dataset, generated below. Visualize the data using the provided `plotter(x, labels)` function.

In [2]:
```
## DO NOT MODIFY
x, labels = make_blobs(n_samples = 1000, cluster_std=[1.0, 2.5, 0.5], random_state = 170)
```

In [3]:
```
## YOUR CODE GOES HERE
plotter(x,labels)
```

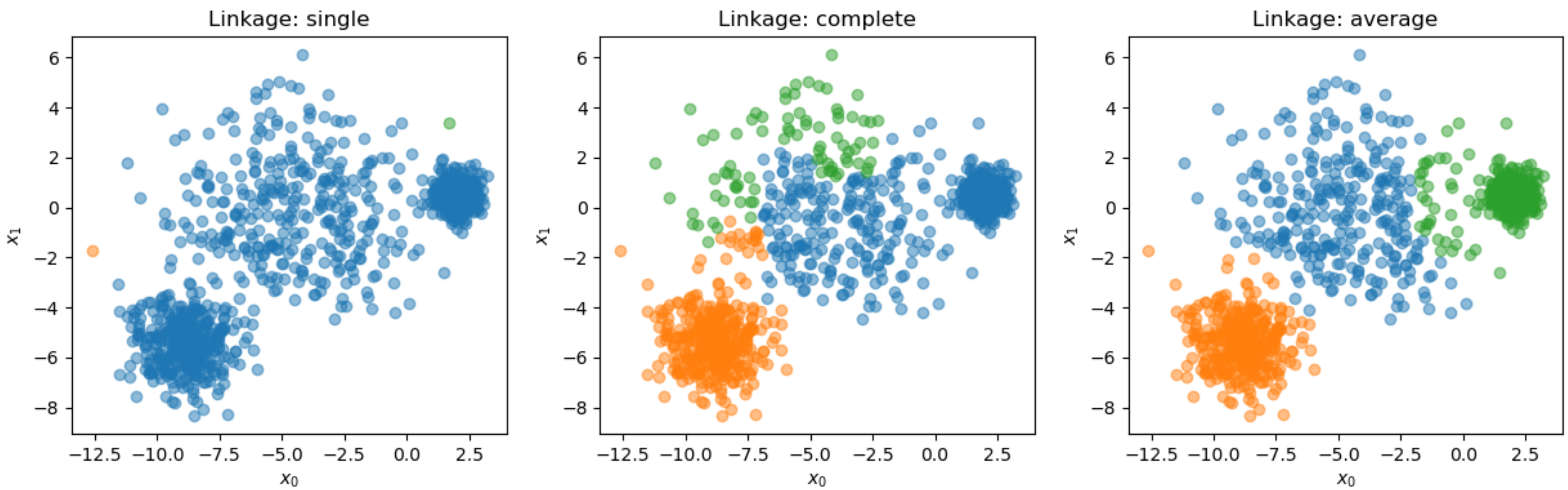


Using the `AgglomerativeClustering()` function, generate 3 side-by-side plots using `plt.subplots()` and the provided `plotter(x, labels, ax, title)` function to visualize the results of the following three linkage criteria `['single', 'complete', 'average']`.

Note: the `plt.subplots()` function will return `fig, ax`, where `ax` is an array of all the subplot axes in the figure. Each individual subplot can be accessed with `ax[i]` which you can then pass to the `plotter()` function's `ax` argument.

In [4]:
```
## YOUR CODE GOES HERE

linkages = ['single', 'complete', 'average']
```

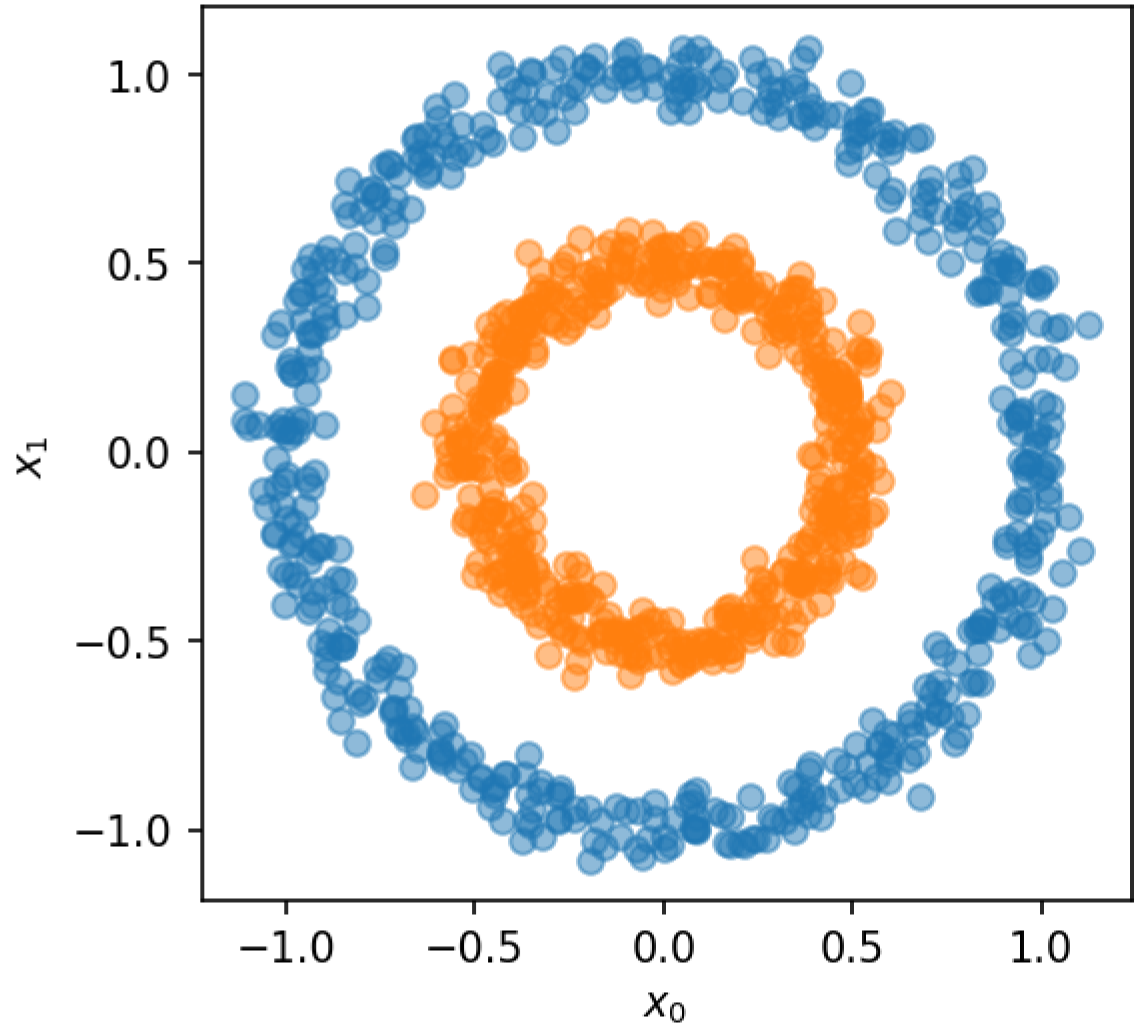
```
fig, ax = plt.subplots(1, 3, figsize = (15,4))
for i,linkage in enumerate(linkages):
    clustering = AgglomerativeClustering(linkage=linkage, n_clusters = 3)
    labels = clustering.fit_predict(x)
    plotter(x,labels=labels,ax=ax[i],title = f'Linkage: {linkage}')
plt.show()
```



Now we will work on the concentric circle dataset, generated below. Visualize the data using the provided `plotter(x, labels)` function.

In [5]:
```
## DO NOT MODIFY
x, labels = make_circles(1000, factor = 0.5, noise = 0.05, random_state = 0)
```

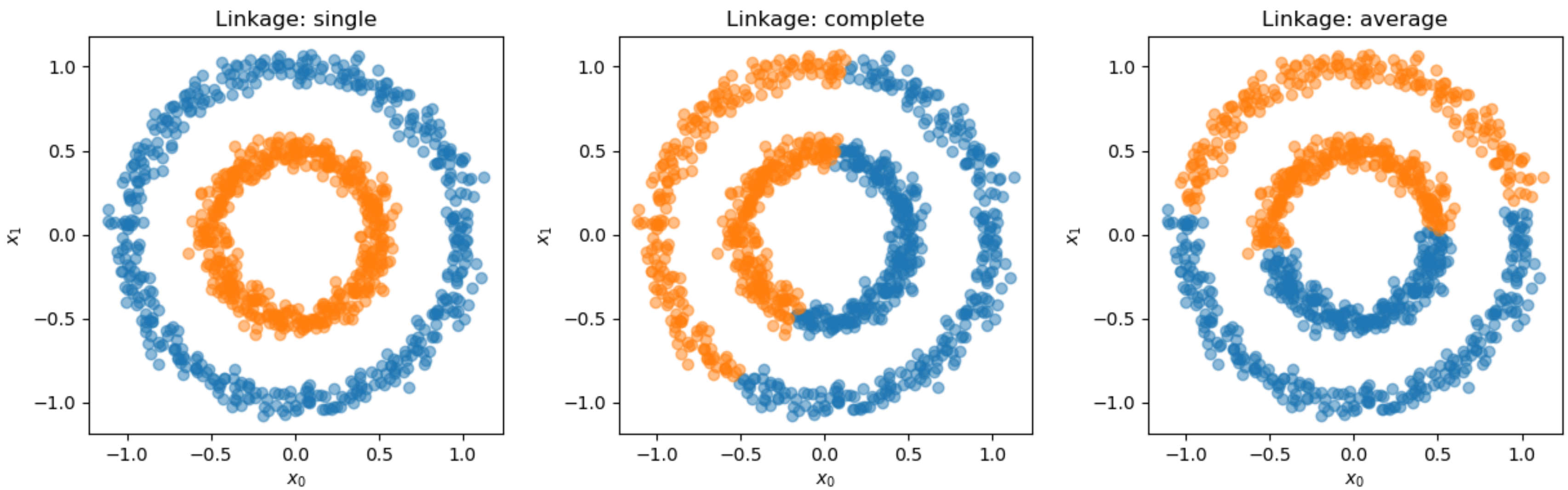
In [6]:
```
## YOUR CODE GOES HERE
plotter(x,labels)
```



Again, use the `AgglomerativeClustering()` function to generate 3 side-by-side plots using `plt.subplots()` and the provided `plotter(x, labels, ax, title)` function to visualize the results of the following three linkage criteria `['single', 'complete', 'average']` for the concentric circle dataset.

In [7]:
```
## YOUR CODE GOES HERE
linkages = ['single', 'complete', 'average']

fig, ax = plt.subplots(1, 3, figsize = (15,4))
for i,linkage in enumerate(linkages):
    clustering = AgglomerativeClustering(linkage=linkage, n_clusters = 2)
```

```
    labels = clustering.fit_predict(x)
    plotter(x,labels=labels,ax=ax[i],title = f'Linkage: {linkage}')
plt.show()
```



Discuss the performance of the three different linkage criteria on the "blob" dataset, and then on the concentric circle dataset. Why do some linkage criteria perform better on one dataset, but worse on others?

For the blob dataset the single and complete linkage didn't provide the correct clustering whereas the average linkage worked well in this dataset. It is because in the average linkage the merge occurs between the two clusters whose centroids have the closest distance. In single and complete linkage the sudden changes happen which lead to incorrect clustering.

For the concentric circles dataset, the single linkage worked better than the average and complete linkage. This is because in single linkage, the merge occurs between two clusters whose two closest members have the smallest distance and for this dataset this criteria is evident.

```
In [ ]:
```